Date=13/11/2020
Lecture By=Manish Mahant
Subject ⇒React Component Composition

IN PREVIOUS LECTURE **(QUICK RECAP) Date-12/11/2020**	**In Today's Lecture (Overview)**
What is React??? Hello React Use cases How does React use JavaScript? Setting up your first React app Questions For Self-Practice // CC For The Day	WHAT'S COMPOSITION IN CODE? WHY REACT COMPONENT COMPOSITION? ENTERING REACT COMPONENT COMPOSITION? GENERALIZATION VS. SPECIALIZATION FOR REACT COMPONENTS REACT COMPONENT COMPOSITION BY EXAMPLE DYNAMIC COMPONENT COMPOSITIONS IN REACT Questions For Self-Practice

WHAT'S COMPOSITION IN CODE?

Let's take one step backwards before tackling composition in React. What's composition in general? It's the ingredients and the arrangement of these ingredients to create something bigger out of it. It's the samples in a piece of music that make up a track. It's the fruits that are used for the perfect smoothie. It's the choreography of dancers in a musical. And it's the internals of a function in programming that need to be arranged in a way to get the desired output:

WHY REACT COMPONENT COMPOSITION?

You have seen how multiple functions can be composed together to achieve something bigger. The same applies to HTML elements and beyond to React components too. Let's encounter both, HTML element composition and React component composition, with a form that submits data. In HTML that form element could look like the following:

```
<form action="javascript:onSubmit();">

  <label>

    Your name: <input type="text" value="">

  </label>

  <button type="submit">Send</button>

</form>
```

However, it's not only the form element but all of its other ingredients and their arrangement as well. It's the input field, the button, and the form that contribute to a greater goal: submit data. The example is taken a bit out of context, because the JavaScript function is missing, but not so the following React example. In React, a Form as React component which is rendered within a App component could look like the following:

```
      import React, { useState } from 'react';

      const App = () => {
        const onSubmit = username => console.log(username);

        return <Form onSubmit={onSubmit} />;
      };

      const Form = ({ onSubmit }) => {
        const [username, setUsername] = useState('');

        return (
          <form
            onSubmit={event => {
              onSubmit(username);

              // prevents browser from reloading
              // which is the native browser behavior
              // for a form submit
              event.preventDefault();
            }}
          >
            <label>
              Your name:
              <input
                type="text"
                value={username}
                onChange={event => setUsername(event.target.value)}
              />
            </label>

            <button type="submit">Send</button>
          </form>
        );
      };

      export default App;
```
Now, wherever we use the Form component, we can capture the username of a user. It's identical to the HTML form from before, isn't it? Not really. At the moment, the Form is only capable of doing one thing. We did loose all the benefits from the HTML element composition from before, because we ended up with a specialized Form component. It can be reused anywhere in our React application, but it handles only one case. To make it effortless to see the difference, we would have to rename the Form component:
```
  import React, { useState } from 'react';

  const App = () => {
    const onSubmit = username => console.log(username);

    return <UsernameForm onSubmit={onSubmit} />;
  };

  const UsernameForm = ({ onSubmit }) => {
    const [username, setUsername] = useState('');

    return (
      <form
        onSubmit={event => {
          onSubmit(username);
          event.preventDefault();
        }}
      >
        <label>
```

```
          Your name:
          <input
            type="text"
            value={username}
            onChange={event => setUsername(event.target.value)}
          />
        </label>

        <button type="submit">Send</button>
      </form>
    );
  };

  export default App;
```

ENTERING REACT COMPONENT COMPOSITION?

There is one property (React prop) that helps us out with this dilemma for our React component: **the React children prop**. It's one special prop provided by React to render something within a component whereas the component isn't aware ahead of time what it will be. A basic example may be the following:

```
const Button = ({ onClick, type = 'button', children }) => (
        <button type={type} onClick={onClick}>
          {children}
        </button>
      );
```

The button element becomes a reusable Button component whereas the Button component doesn't know what it renders except for the button. Let's use the children prop for our previous example to substitute our HTML form element with a Form component that renders all its inner content with React's children prop:

```
      const UsernameForm = ({ onSubmit }) => {
        const [username, setUsername] = useState('');

        return (
          <Form
            onSubmit={event => {
```

```
          onSubmit(username);
          event.preventDefault();
        }}
      >
        <label>
          Your name:
          <input
            type="text"
            value={username}
            onChange={event => setUsername(event.target.value)}
          />
        </label>

        <button type="submit">Send</button>
      </Form>
    );
  };

  const Form = ({ onSubmit, children }) => (
    <form onSubmit={onSubmit}>{children}</form>
  );
```

GENERALIZATION VS. SPECIALIZATION FOR REACT COMPONENTS

In our case, we have one specialized Form component (UsernameForm) to capture the username information from a user. However, you could also use the Form component directly in the App component. The App component then makes it a specialized Form component by passing all the displayed information as children and other props to it:

```
  import React, { useState } from 'react';

  const App = () => {
    const onSubmit = username => console.log(username);

    const [username, setUsername] = useState('');
```

```
      return (
        <Form
          onSubmit={event => {
            onSubmit(username);
            event.preventDefault();
          }}
        >
          <InputField value={username} onChange={setUsername}>
            Your name:
          </InputField>

          <Button type="submit">Send</Button>
        </Form>
      );
    };
```

The UsernameForm component disappears. In the App component, you take all the ingredients (e.g. Form, InputField, Button), give them your specialized flavor (e.g. onSubmit, username, setUsername), and arrange them however you want them to appear within the Form component. What you get is a composed Form component that is specialized from the outside (App component). Anyway, you can also keep the UsernameForm, if this kind of specialized Form is used more than once in your application:

```
const App = () => {
      return (
        <div>
          <UsernameForm onSubmit={username => console.log(username)} />
          <UsernameForm onSubmit={username => console.log(username)} />
        </div>
      );
    };

    const UsernameForm = ({ onSubmit }) => {
      const [username, setUsername] = useState('');

      return (
        <Form
          onSubmit={event => {
            onSubmit(username);
            event.preventDefault();
          }}
        >
          <InputField value={username} onChange={setUsername}>
            Your name:
          </InputField>

          <Button type="submit">Send</Button>
        </Form>
      );
```

REACT COMPONENT COMPOSITION BY EXAMPLE

You have seen how component composition is mainly used for reusable React components that require a well-designed API. Often you will find this kind of component composition just to layout your application as well. For instance, a SplitPane component, where you want to show something at the left and the right as the inner content of the component, could make use of React props to render more than one child component:

```
      const SplitPane = ({ left, right }) => (
        <div>
          <div className="left-pane">{left}</div>
          <div className="right-pane">{right}</div>
        </div>
      );
```

Then it could be used the following way in another React component whereas you decide what you render as children in which of both slots:

```
<SplitPane
  left={
    <div>
      <ul>
        <li>
          <a href="#">Link 1</a>
        </li>
        <li>
          <a href="#">Link 2</a>
        </li>
      </ul>
    </div>
  }
  right={<Copyright label="Robin" />}
/>
```

DYNAMIC COMPONENT COMPOSITIONS IN REACT

Often you see something like the following App component whereas React Router is used to compose dynamic components, depending on the selected route (URL), into the Route components:

```
      import React from 'react';
      import {
        BrowserRouter as Router,
        Route,
      } from 'react-router-dom';

      import Navigation from './Navigation';
      import LandingPage from './Landing';
      import SignUpPage from './SignUp';
      import SignInPage from './SignIn';

      const App = () => (
        <Router>
          <div>
            <Navigation />

            <hr />

            <Route exact path='/' component={LandingPage} />
            <Route path='/register' component={SignUpPage} />
            <Route path='/login' component={SignInPage} />

            <Footer />
          </div>
        </Router>
      );
```

Click Here TO Know More about It
https://www.robinwieruch.de/react-component-composition

Questions For Self-Practice

CC
https://au-assignment.s3.ap-south-1.amazonaws.com/Week_20_Day_5_Challenge-b05c2922-b10a-4820-bb70-cfff8050d81c.pdf

Assignment
https://au-assignment.s3.ap-south-1.amazonaws.com/Week_20_Day_5_Assignment-7a5b6a92-6f41-4bfd-bc52-17a53664f736.pdf